IMPLEMENTATION OF SOURCE CODING TECHNIQUES USING LABVIEW

- Group No. N8_3
- 32463 TEJAS BORA
- 32464 VAIBHAV THAKUR
- 32466 VYENKATESH PAREEK

OUTLINE

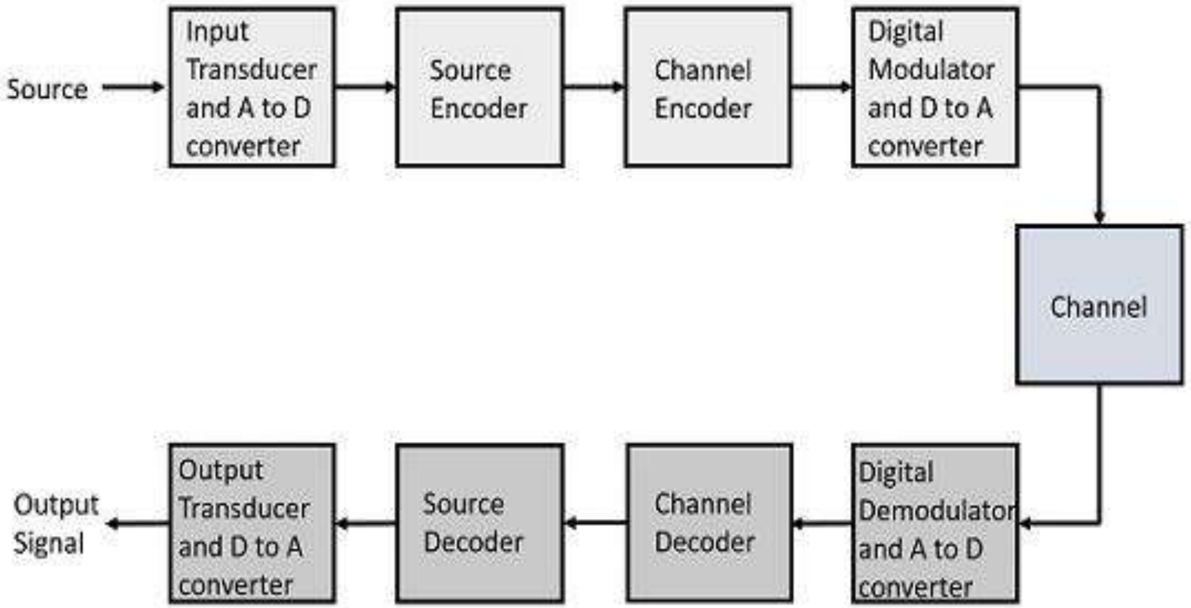
- Introduction
- Block Diagram
- Algorithms
- Flowcharts
- Simulation
- Applications
- Future Scope

Introduction

- What is Source Coding and why is it necessary?
- Source coding (source compression coding) The use of variable-length codes in order to reduce the number of symbols in a message to the minimum necessary to represent the information in the message, or at least to go some way toward this, for a given size of alphabet. In source coding the particular code to be used is chosen to match the source (i.e. the relative probabilities of the symbols in the source alphabet) rather than any channel through which the message may ultimately be passed.
- The main problem in source coding is to ensure that the most probable source symbols are represented by the shortest codewords, and the less probable by longer codewords as necessary, the weighted average codeword length being minimized within the bounds of Kraft's inequality. The most widely used methods for ensuring this are Huffman coding and Shannon–Fano coding; the former is more efficient for a given extension of the source but the latter is computationally simpler. In either case, a large extension of the source may be necessary to approach the limiting compression factor given by the source coding theorem.

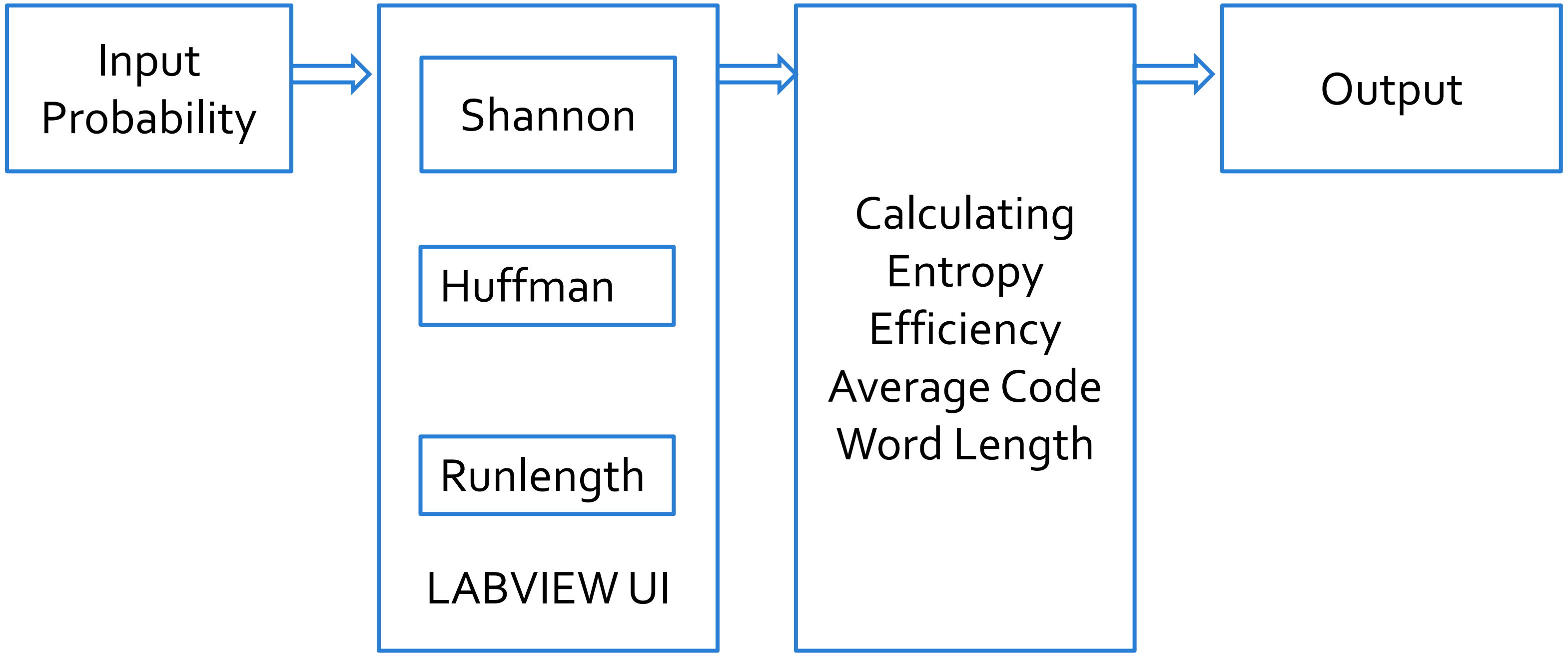
Block Diagram

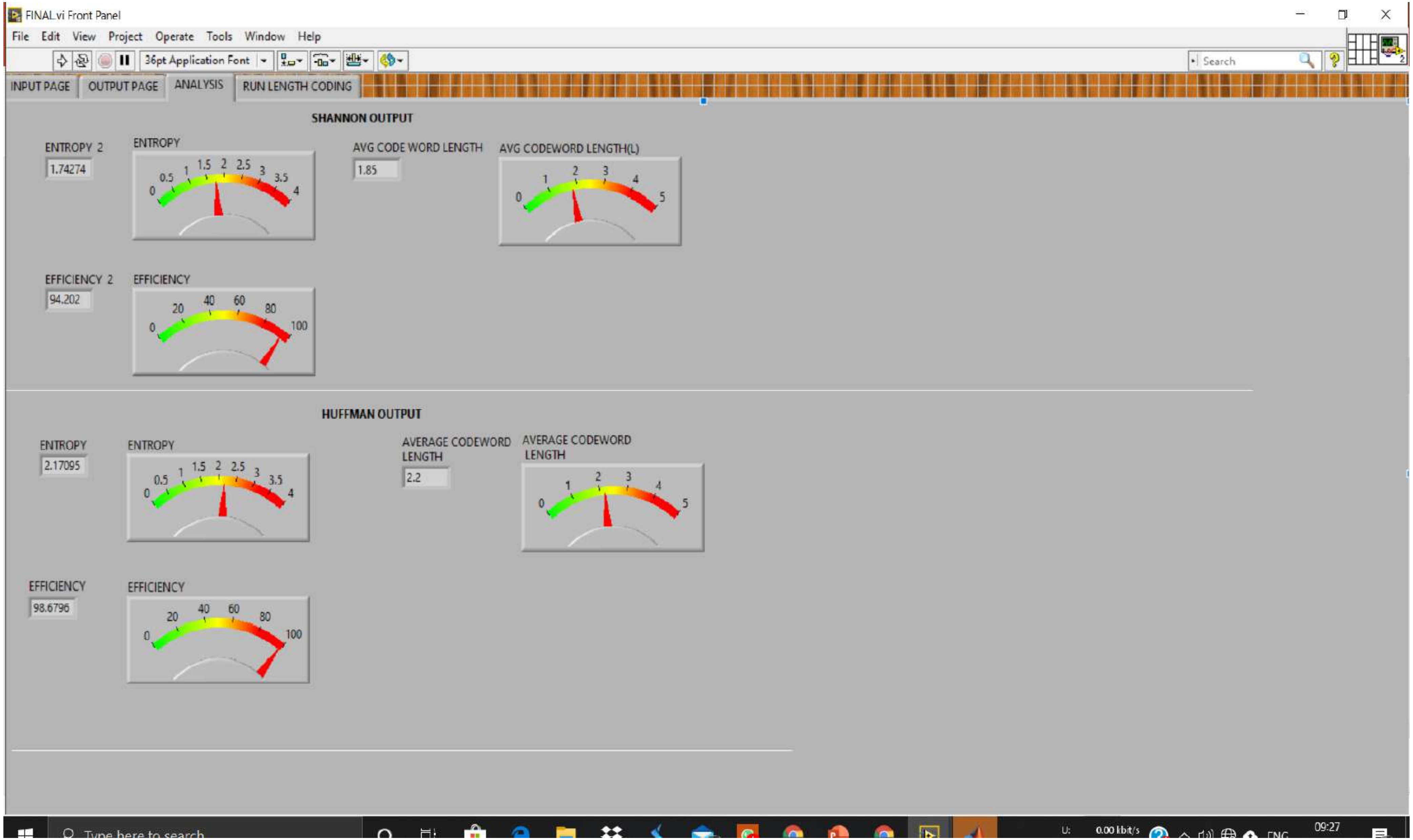
- The block diagram shown here is of a basic digital communication system,it shows the importance and significance of source coding in communication system



Basic Elements of a Digital Communication System

Block Diagram





FINAL.vi Front Panel
File Edit View Project Operate Tools Window Help
36pt Application Font
INPUT PAGE
OUTPUT PAGE
ANALYSIS
RUN LENGTH CODING
SHANNON OUTPUT
ENTROPY 2
1.74274
ENTROPY
AVG CODE WORD LENGTH
1.85
AVG CODEWORD LENGTH(L)
EFFICIENCY 2
94.202
EFFICIENCY
HUFFMAN OUTPUT
ENTROPY
2.17095
ENTROPY
AVERAGE CODEWORD LENGTH
2.2
AVERAGE CODEWORD LENGTH
EFFICIENCY
98.6796
EFFICIENCY

Algorithm - Shannon

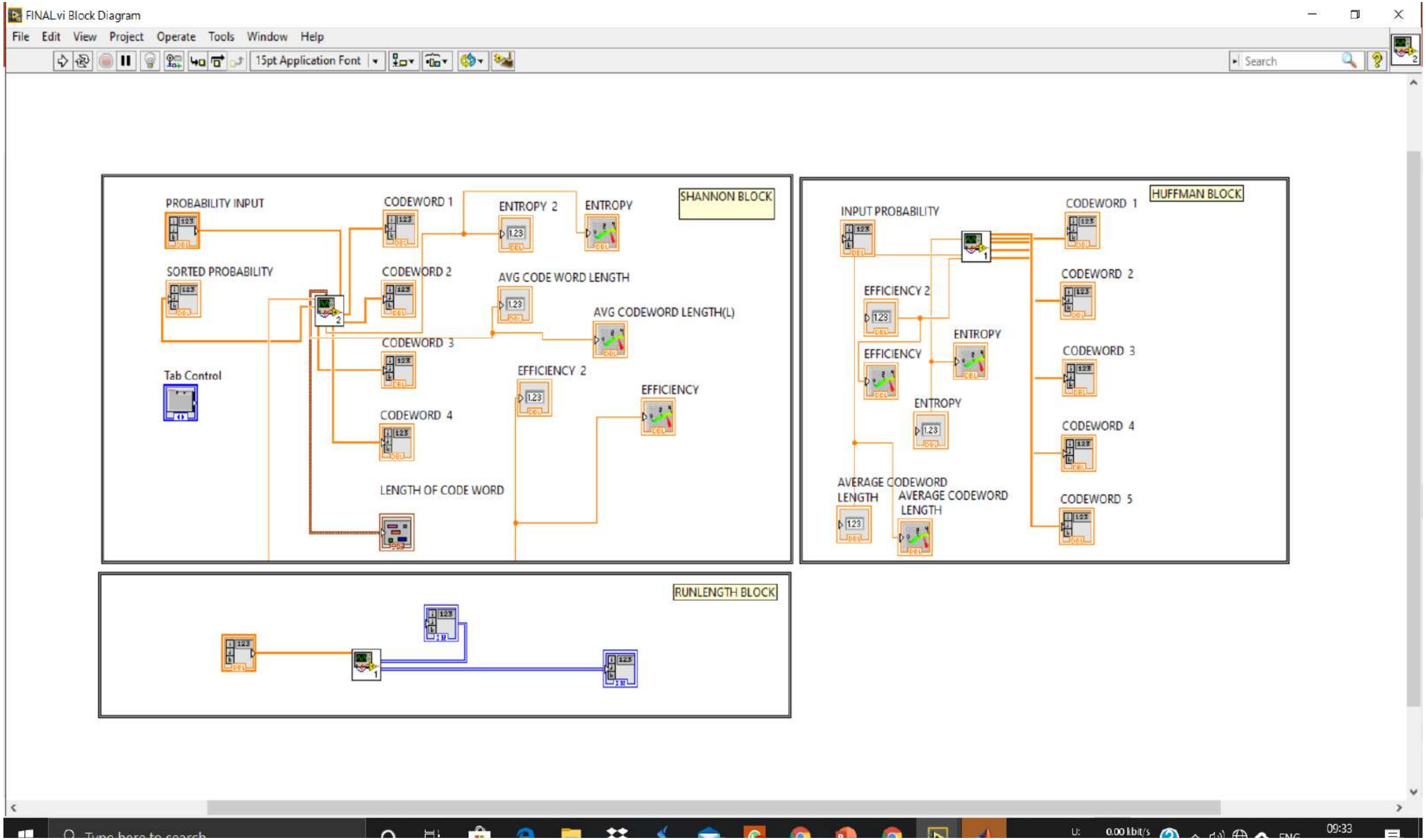
- 1. For a given list of symbols, develop a corresponding list of probabilities or frequency counts so that each symbol's relative frequency of occurrence is known.
- 2. Sort the lists of symbols according to frequency, with the most frequently occurring symbols at the left and the least common at the right.
- 3. Divide the list into two parts, with the total frequency counts of the left part being as close to the total of the right as possible.
- 4. The left part of the list is assigned the binary digit 0, and the right part is assigned the digit 1. This means that the codes for the symbols in the first part will all start with 0, and the codes in the second part will all start with 1.
- 5. Recursively apply the steps 3 and 4 to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree

Algorithm - Huffman

- Calculate the frequency of each character in the string.
- Sort the characters in increasing order of the frequency. These are stored in a priority
- Make each unique character as a leaf node.
- Create an empty node z. Assign the minimum frequency to the left child of z and assign the second minimum frequency to the right child of z. Set the value of the z as the sum of the above two minimum frequencies.
- Remove these two minimum frequencies from Q and add the sum into the list of frequencies (* denote the internal nodes in the figure above).
- Insert node z into the tree.
- Repeat steps 3 to 5 for all the characters.
- For each non-leaf node, assign 0 to the left edge and 1 to the right edge.

Algorithm - Runlength

- Pick the first character from source string.
- Append the picked character to the destination string.
- Count the number of subsequent occurrences of the picked character and append the count to destination string.
- Pick the next character and repeat steps 2) 3) and 4) if end of string is NOT reached.


FINAL.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
SHANNON BLOCK
PROBABILITY INPUT
SORTED PROBABILITY
Tab Control
CODEWORD 1
CODEWORD 2
CODEWORD 3
CODEWORD 4
ENTROPY 2
ENTROPY
AVG CODE WORD LENGTH
AVG CODEWORD LENGTH(L)
EFFICIENCY 2
EFFICIENCY
LENGTH OF CODE WORD
HUFFMAN BLOCK
INPUT PROBABILITY
EFFICIENCY 2
EFFICIENCY
ENTROPY
ENTROPY
AVERAGE CODEWORD LENGTH
AVERAGE CODEWORD LENGTH
CODEWORD 1
CODEWORD 2
CODEWORD 3
CODEWORD 4
CODEWORD 5
RUNLENGTH BLOCK

Flowchart - Shannon

START

ARRANGE PROBABILITIES IN DECREASING ORDER

DIVIDE IN GROUPS SUCH AS PROBABILITIES ARE EQUALLY DIVIDED

ASSIGN 1 TO ALL MEMBERS OF ONE GROUP AND 0 TO ALL MEMBERS OF SECOND GROUP

REPEAT UNTIL NO SOURCE PROBABILITY IS LEFT

END

Flowchart - Huffman

START

ARRANGE IN DECREASING ORDER OF PROBABILITIES

MERGE LAST TWO SYMBOLS AND ASSIGN 1 AND 0 TO THEM

AGAIN ARRANGE IN DECREASING ORDER REPLACING THE LAST TWO SYMBOLS WITH MERGED ONE

REPEAT UNTIL NO UNMERGED NODE IS LEFT

READ FROM BRANCH TO TREE TO GET CODEWORD

END

Flowchart - Runlength

START

READ A SOURCE STRING

READ A CHARACTER FROM SOURCE STRING

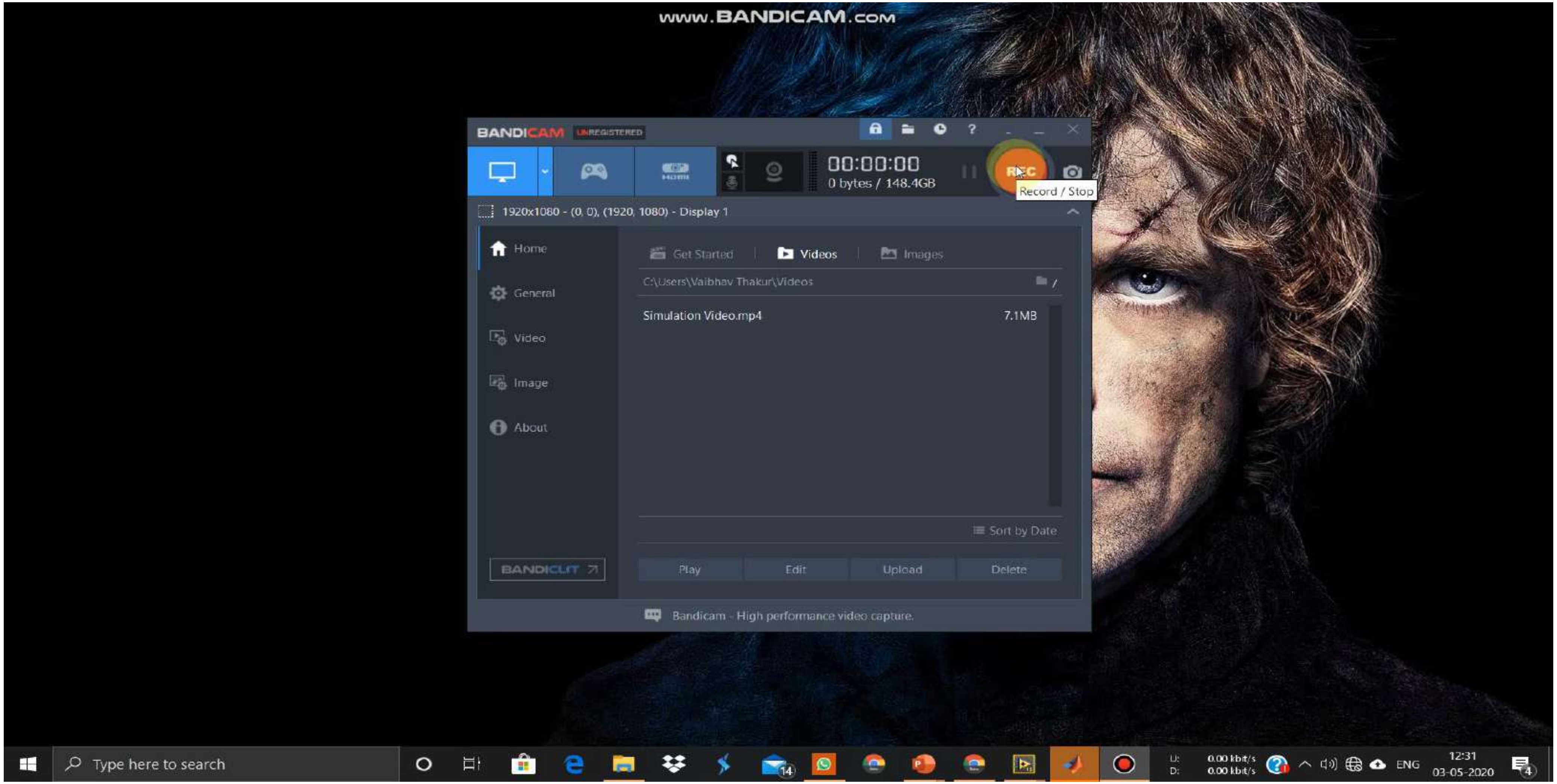
SCAN NEXT CHARACTER , TILL IT IS SAME AS PREVIOUS KEEP UPDATING COUNT

SEND THE CHARACTER FOLLOWED BY COUNT

REPEAT TILL LAST CHARACTER OF SOURCE STRING

END

Simulation Video of Project



Applications

- Data Compression
- Image Processing
- Communication Systems
- Cryptography

Future Scope

- More focus can be on improvement of compression ratio.
- New methods can be proposed to decrease time complexity
- Experiments can be done on 3D video files using proposed technique.
- Applying developed methods on large dataset could be a subject for future research.